A2Q1: SparseMatMult

In [1]:
```
import numpy as np
from scipy.sparse import dok_matrix
from copy import deepcopy
import matplotlib.pyplot as plt
```

In [2]:
```
def SparseMatMult(G, x):
    '''
      y = SparseMatMult(G, x)

      Multiplies a vector (x) by a sparse matrix G,
      such that y = G @ x .

      Inputs:
        G is an NxM dictionary-of-keys (dok) sparse matrix
        x is an M-vector

      Output:
        y is an N-vector
    '''
    rows,cols = G.nonzero()
    Nrows,Ncols = np.shape(G)
    y = np.zeros(Nrows)

    for i in range(0, len(rows)):
        y[rows[i]] = y[rows[i]] + G[rows[i],cols[i]] * x[cols[i]][0]
    yt = [[y[i]] for i in range(Nrows)]
    return yt
```

In [3]:
```
# A simple test
G = dok_matrix((3,4), dtype=np.float32)
G[0,0] = 3
G[0,3] = 1
G[1,1] = 4
G[2,0] = 5
G[2,1] = 6
x = np.array([[1], [2], [3], [4]])

y = SparseMatMult(G, x)
print(y)
```

```
[[7.0], [8.0], [17.0]]
```

A2Q2: Page Rank

In [64]:
```
def PageRank(G, alpha):
    '''
     p, iters = PageRank(G, alpha)

     Computes the Google Page-rank for the network in the adjacency matrix G.

     Note: This function never forms a full RxR matrix, where R is the number
           of node in the network.

     Input
```

```
        G     is an RxR adjacency matrix, G[i,j] = 1 iff node j projects to node i
              Note: G must be a dictionary-of-keys (dok) sparse matrix
        alpha is a scalar between 0 and 1

      Output
        p     is a probability vector containing the Page-rank of each node
        iters is the number of iterations used to achieve a change tolerance
              of 1e-8 (changes to elements of p are all smaller than 1e-8)

      [-1] if code is not readable
    '''

    R = np.shape(G)[0]  # R = Number of nodes
    p = [[0] for i in range(R)]
    iters = -1
    for i in range(0, R):
        p[i][0] = 1 / R
    P = dok_matrix((R,R), dtype=np.float32)
    for i in range(0, R):
        for j in range(0, R):
            if G[i,j] != 0:
                P[i,j] = G[i,j] / np.sum(G[:,i])
    e = [[1] for i in range(R)]
    dt = [0 for i in range(R)]
    for i in range(R):
        if len(np.nonzero(G[:,i])[0]) == 0:
            dt[i] = 1
    tolerance = 0.00000001
    terminate = False

    while terminate == False:
        terminate = True
        pp = np.dot(alpha, SparseMatMult(P,p)) + np.dot(alpha/R*np.matmul(dt, p)[0], e)
        for i in range(R):
            if (np.absolute(p[i] - pp[i]) > tolerance):
                terminate = False
        p = pp
        iters = iters + 1

    return p, iters
```

In [65]:
```
# A simple test
G = np.array([[0,0,0,0],[1,0,0,1],[0,1,0,1],[0,1,1,0]])
p, iters = PageRank(G, 0.1)
print(p, iters)
```

```
[[0.225     ]
 [0.24880952]
 [0.275     ]
 [0.25119048]] 5
```

A2Q3: Illegal Trading Network

(a) Create sparse matrix

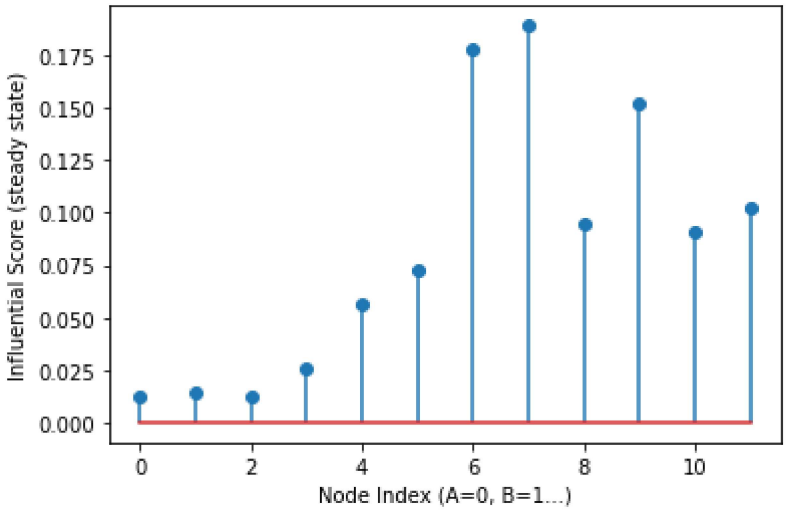
In [66]:
```
G = dok_matrix((12,12), dtype=np.float32)
G[0,1] = 6
G[0,2] = 47
```

```
G[0,4] = 9
G[1,0] = 38
G[1,2] = 29
G[1,5] = 9
G[2,0] = 38
G[2,1] = 41
G[2,3] = 8
G[3,2] = 24
G[3,5] = 28
G[3,4] = 4
G[4,0] = 24
G[4,3] = 42
G[4,5] = 19
G[4,11] = 6
G[4,6] = 13
G[5,1] = 53
G[5,3] = 50
G[5,4] = 9
G[5,11] = 18
G[5,7] = 15
G[6,4] = 39
G[6,11] = 47
G[6,7] = 21
G[6,9] = 30
G[6,8] = 24
G[7,5] = 22
G[7,11] = 29
G[7,6] = 17
G[7,8] = 24
G[7,10] = 33
G[7,9] = 40
G[8,6] = 23
G[8,7] = 21
G[8,9] = 5
G[8,10] = 7
G[9,7] = 10
G[9,6] = 27
G[9,8] = 32
G[9,10] = 60
G[10,8] = 20
G[10,7] = 18
G[10,9] = 25
G[11,4] = 39
G[11,6] = 20
G[11,5] = 22
G[11,7] = 15
```

(b) Run PageRank on netork

In [68]:

```
p, iters = PageRank(G, 1)
plt.stem(p);
plt.xlabel('Node Index (A=0, B=1...)')
plt.ylabel('Influential Score (steady state)');
```



(c) Note to police

Node H is the most influential.